09/138,456. The client 52 and the pipelines 55-59 will be described in more detail hereinafter.

Each frame buffer 65-69 outputs a stream of graphical data to the compositor 76. The compositor 76 is configured to combine or composite each of the data streams from frame buffers 65-69 into a single data stream that is provided to display device 83, which may be a monitor (*e.g.*, cathode ray tube) or other device for displaying an image. The graphical data provided to the display device 83 by the compositor 76 defines the image to be displayed by the display device 83 and is based on the graphical data received from frame buffers 65-69. The compositor 76 will be further described in more detail hereinafter. Note that each data stream depicted in FIG. 3 may be either a serial data stream or a parallel data stream.

In the preferred embodiment, the client 52 and each of the pipelines 55-59 are respectively implemented via stand alone computer systems, commonly referred to as a "computer workstations." Thus, the system 50 shown by FIG. 3 may be implemented via six computer workstations (*i.e.*, one computer workstation for the client 52 and one computer workstation for each of the pipelines 55-59). However, it is possible to implement the client 52 and pipelines 55-59 via other configurations, including other numbers of computer workstations or no computer workstations. As an example, the client 52 and the master pipeline 55 may be implemented via a single computer workstation. Any computer workstation used to implement the client 52 and/or pipelines 55-59 may be utilized to perform other desired functionality when the workstation is not being used to render graphical data.

Furthermore, as shown by FIG. 3, the client 52 and the pipelines 55-59 may be interconnected via a local area network (LAN) 62. However, it is possible to utilize other

types of interconnection circuitry without departing from the principles of the illustrated system.

FIG. 4 depicts a more detailed view of the client 52. As can be seen by referring to FIG. 4, the client 52 preferably stores the graphics application 17 in memory 102. Through conventional techniques, the application 17 is executed by an operating system 105 and one or more conventional processing elements 111, such as a central processing unit (CPU), for example. The operating system 105 performs functionality similar to conventional operating systems. More specifically, the operating system 105 controls the resources of the client 52 through conventional techniques and interfaces the instructions of the application 17 with the processing element 111 as necessary to enable the application 17 to run properly.

The processing element 111 communicates to and drives the other elements within the client 52 via a local interface 113, which can include one or more buses. Furthermore, an input device 115, for example, a keyboard or a mouse, can be used to input data from a user of the client 52, and an output device 117, for example, a display device or a printer, can be used to output data to the user. A disk storage mechanism 122 can be connected to the local interface 113 to transfer data to and from a nonvolatile disk (*e.g.*, magnetic, optical, *etc.*). The client 52 is preferably connected to a LAN interface 126 that allows the client 52 to exchange data with the LAN 62.

In the preferred embodiment, X Protocol is generally utilized to render 2D graphical data, and OpenGL Protocol (OGL) is generally utilized to render 3D graphical data, although other types of protocols may be utilized in other embodiments. By way of background, OpenGL Protocol is a standard application programmer's interface (API) to hardware that accelerates 3D graphics operations. Although OpenGL Protocol is designed to be window system independent, it is often used with window systems, such as

the X Window System, for example. In order that OpenGL Protocol may be used in an X Window System environment, an extension of the X Window System has been developed called GLX. For more complete information on the GLX extension to the X Window System and on how OpenGL Protocol can be integrated with the X Window System, see for example Mark J. Kilgard, *OpenGL Programming for the X Window System* (Addison-Wesley Developers Press 1996), which is incorporated herein by reference.

When the application 17 issues a graphical command, a client side GLX layer 131 of the client 52 transmits the command over LAN 62 to master pipeline 55. FIG. 5 depicts a more detailed view of the master pipeline 55. Similar to client 52, the master pipeline 55 includes one or more processing elements 141 that communicate to and drive the other elements within the master pipeline 55 via a local interface 143, which can include one or more buses. Furthermore, an input device 145, for example, a keyboard or a mouse, can be used to input data from a user of the pipeline 55, and an output device 147, for example, a display device or a printer, can be used to output data to the user. A disk storage mechanism 152 can be connected to the local interface 143 to transfer data to and from a nonvolatile disk (*e.g.*, magnetic, optical, *etc.*). The pipeline 55 may be connected to a LAN interface 156 that allows the pipeline 55 to exchange data with the LAN 62.

The pipeline 55 also includes an X server 162. The X server 162 may be implemented in software, hardware, or a combination thereof, and in the embodiment shown by FIG. 5, the X server 162 is implemented in software and stored in memory 164. In the preferred embodiment, the X server 162 renders 2D X window commands, such as commands to create or move an X window. In this regard, an X server dispatch layer 173 is designed to route received commands to a device independent layer (DIX) 175 or to a GLX layer 177. An X window command that does not include 3D data is interfaced with DIX, whereas an X window command that does include 3D data (*e.g.*, an X command having